Computational Geometry

Prof. Sandeep Sen

Department of Computer Science and Engineering

Indian Institute of Technology, Delhi

Module No. # 01

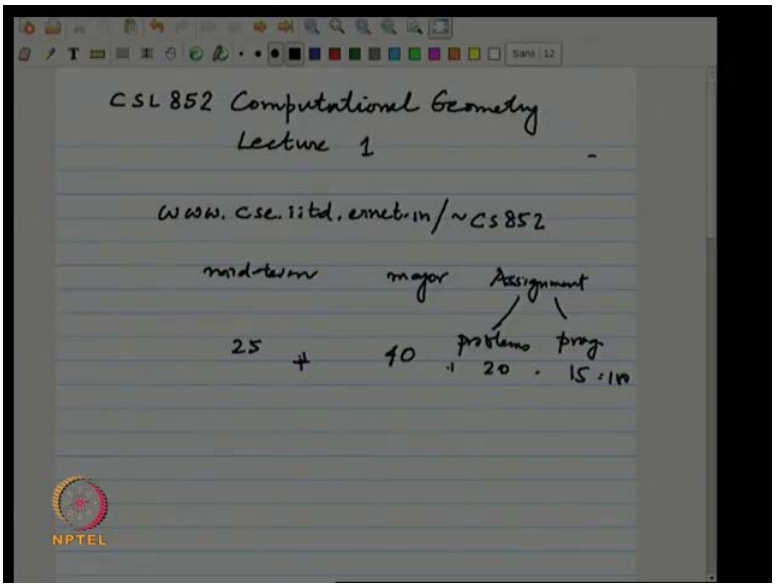
Introducing Using Basic Visibility Problems

Lecture No. # 01

Introduction

We will come to the course on computational geometry this is semester one 2010/ 2011. So, the first thing I should mention is what I expect from you as pre requisites.

(Refer Slide Time: 00:34)



So, I expect that you would have done a course in algorithms, and proceeded by that you should have done I am sure some course in data structures, and it will be nice if you have also done a course in discrete structures. Discrete structures essentially meaning commentaries. So, I am not going to cover the basics of data structures or basic algorithmic paradigms like whatever divide and concord recurrence. So, I expect you to be familiar with those tools and techniques and because you will write from let us say may be today or tomorrow itself you will see that momentary try to develop any algorithms you now we will be certainly relying on these well known techniques.

And over the course, we will also learn about some other techniques that you may not have encountered before. In fact, I would say that computation geometry I will give you brief history. So, the word computational geometry is it was coined sometime in the mid seventies, but even prior to that there was work in geometric algorithms especially in the context of some very classical problems like minimum spanning tree, traveling sales man problem a standard trees. So, normally you get to know about this problems when you are doing a course in algorithms or a graph algorithms to be more precise, but after this, some of these algorithms were initially defined or described in the context of when the setting is Euclidian space rather than the graph.

So, they have very natural applications when you know of problems like Euclidian minimum spanning tree, Euclidian traveling sales man problem and so and so forth, and when you actually try to solve this problems in graph that is modeled the problems in graphs you actually lose some of the structure.

So, that is you lose the basic geometry of the properties of the Euclidian space and there by actually the although the problem becomes more general I mean you take it to graphs because you do not have things like triangular inequalities etcetera, but on the other hand because you abstract the problem you know if may be its applicable to many other settings as well.

But then if you are actually dealing with a problem that is arising in Euclidian space, then you may want to actually also exploit some of the properties of the Euclidian space. So, in this course, we will be talking about problems that occurred naturally in the Euclidian space. So, when we develop or design algorithms, we will be using those facts and also the techniques that we have learnt in algorithms design in other courses; those will come in handy.

Computational geometry can we thought about as more like geometry computational, that may be the right term, but this term was coined by the persons as in the mid seventies. In fact, the first textbook that was written I think kind of standardized term. So, I have mentioned on my web page. So, this is one of the course pages that you may want to refer to from time to time.

So, that is the web page that I maintain and then not sure there is a link basically I think it is cs852. So, from this page you can open that link and then what is prescribed in the

are some reference textbooks, the course outline and from time to time we will put up assignment sheets and also in this course I, towards the latter half, not the first half, will actually assign some give me some programming assignments. So, I would like you to actually also get some experience in actually writing programs to solve geometric problems.

It is going to be somewhat different and more challenging than some of the algorithms you have implemented in other courses, especially you may have implemented search algorithms, standard computer algorithms like sorting, some graph algorithms, m s t.

The main difference between implementing a geometric algorithm (( )) this cemetery algorithms is that you are dealing with points in Euclidian space when therefore; you have dealing with real numbers. So, once you deal with real numbers which are actually very unreal because the computers do not have real numbers, then you have to struggle with issues like over flow, under flow, stability; all those things, the things that you normally learn in numerical analysis course, something that will encounter when you try to actually implement the algorithms that we will design in the developing the class.

And to tackle those problems, you know of to some handle those problems, you will need to think beyond just the (( )) aspects of the algorithms, but also see how the points will be represented in, what kind of precision we need etcetera. So, those also will become issues and these are serious issues when people try to implement geometrical algorithms.

In this course, we are not we are going to kind of avoid those issues for most parts, may be once I will maybe elsewhere a lecture to just trying to give you an over view of what are the kind of problems that arise when you implement geometric algorithms, but the theory of actually trying to implement algorithms in a way where you minimize errors, I said is very similar to numerical computing and that requires completely separate discussions. In fact, I would go as far as to say although this area is about a that a more than thirty years old now, but I think people are still trying to come up with a clean model of implementing geometric algorithms.

So, it is very much a research topic you know how to do the clean implementation given that you have to handle real numbers. Anyway so, outline of what I expect from you in terms of assignment. So, there will be two or three written assignments and I would say

one or two programming assignments depending you know I have not decided yet how big, but they will take some time. I will probably limit myself to one midterm exam rather two minus and we will have a major exam. So, I haven't quite given much thought to how the marks we will distributed, but let us say, I am I am just saying, let us say. So, have minors sorry let say midterm actually not a minor, in the midterm you have a major, you have assignments. So, there are three components and I am saying assignments, there will be some basic theoretical assignments in problems and some programming assignments. So, let see. So, maybe I will do something like perhaps something like this looks (( )) let us say, twenty-five, forty and may be twenty and fifteen; that decides up to two hundred. So, let say, this is about the rough distribution that that we have in the course. I have described the syllabus in details in the in the course page. I do not want to go through it because most of the terms would not make sense today. Rather I would give you start with an example of a geometric problem. Also let me also clarify that I said computation geometry is about designing and implementing algorithms for problems that are geometric in nature and that is quite distinct from trying to prove geometric theorems automatically.

So, that as that is completely different area. That is done by also there is a community of people doing that who do research in theorem proving and actually they have done impressive work, but like any other theorem proving exercise, this actually trying to prove a new theorem is extremely hard and certainly not feasible algorithmically for most of the algorithms. So, people are struggling with that, but just to even get that frame work going you know it is a very impressive achievements, but that is done again by I am saying that people in the theorem proving community.

So, we are not going to try to prove theorems using computers. We will try to prove the theorems by hand. Whatever we prove about the algorithms are the running time the analysis we will do by hand.

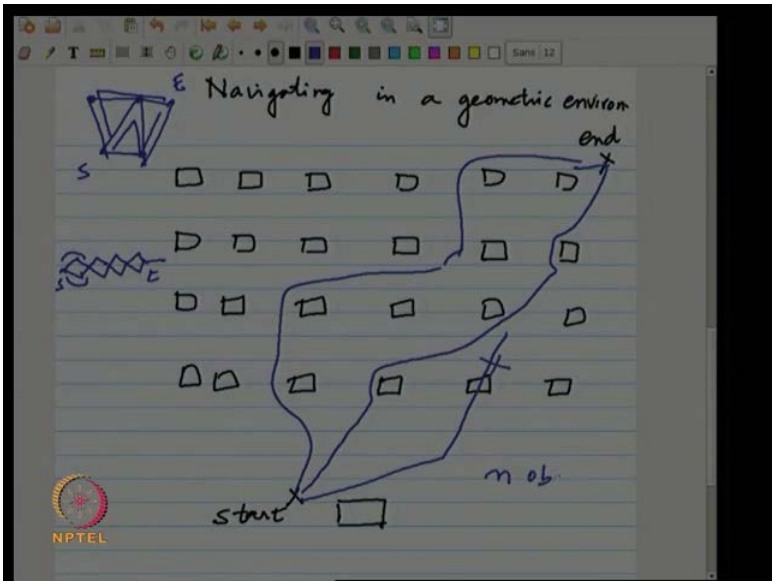
Feel free to ask me any questions if I miss something that is very relevant. So, let me proceed with an example to illustrate in what kind of what geometric problems mean to us in what kind of techniques people try to use. So, let me take up a very natural example like which is essentially a navigating. So, navigating in a geometric environment which is basically that you have given. So, you can think about this room as a geometric environment where there are all of us can be describe with some geometry primitives.

So, the furniture, the people, the cameras; whatever. So, this entire that geometric description of this room is what a mean the geometric environment and then we may want to move some object from one part of the room to the other part. That is what a mean by navigation.

And of course, this navigation has to be done in a way such that it is not allowed to pass through solid objects. So, you can pass through the air, but you cannot pass through a solid object. So, you cannot. So, if you collide, you can only set of at best you have to circumvent that object.

So, to make things easy and for also you know from the point of view of we have to draw. So, let me draw the 2 dimensional log of this. So, suppose I take the projection of this room on the plane. So, what does it look like?

(Refer Slide Time: 12:17)



So, it could look like. So, it could look like, we have his people sitting on this like this here, some rows of this here, again other row and here is where I am seated and we want to move let us say, let us say to begin with, it is just a point, which is easier because it has zero width. So, suppose I want to move a point and it may indicate that by cross, from this point to let us say to this corner of the row. This is the starting point start and this is the end. So, this is the game well. So, there are various ways of doing this. Let me choose different color. So, start walking, start walking you I cannot I have to way around this. So, I go, I go here may be I have to go around this. Alright I found a path.

So, this is how I have navigated from the start to end. Someone else we decide to do in some other way. So, here, there we will go, that is also a legal path. Someone more may want to do this way and this is not a legal path. It just crashes into furnaces, we cannot do that, one has to go around.

So, what is observed about these paths is that you can have several ways of navigating from this starting point to the end point without colliding into the obstacles, but which is the way that you are going to choose? Suppose you know I have write a program. So, I can certainly describe these objects in some way, these are let us say for simplicity all these are rectangles. So, how do I move this point and how do I write a program that moves this point from the starting to the end point that goes around obstacles. So, on what on it is to choose a random path, even to choose a random path I have to ensure in the minimum that it does not collide with the with the obstacles.

So, somehow we need to capture that fact and the one difficulty or let us say one initial difficulty is that the possible number of paths seems to be just too large. In fact, the possible numbers of paths that will infinite because I can take any legal path here and may just decide to just perturb this little bit just move it like this and that is it. So, that is another legal path. So, with every small perturbation of one path, I can get another legal path. So, I am dealing with an infinite number of paths and this is what sets it to path from a path problem in graphs.

So, if I have a graph. So, suppose I have a graph like... So, I could have lots of edges in this graph does not matter I can have lots of edges this. This starting point and this is the ending point, but still the number of different ways I can move from the starting to the end point is finite. So, I could do like this, I could go like this, but again it just a infinite number and how a large it is it is till finite.

It may be exponential that it suppose I have n vertices there could be you know it could have you know this is a very common example to show that the number of paths could be exponential.

So, you could have these choices at every point, this is the starting point, this is the end point. Every point you have two possibilities to pursue and in if there are n such rhombus, you basically have 2 to the power n possibilities to provides, but still it is a finite number, it’s exponential, but still finite.

Here if a parameter is my problem by seeing that have n obstacles, I have n obstacles, I were knew obstacles, I cannot say that I have less than 2 d power n paths. It is not true is not true in the case of geometry.

So, the first difficulty that we will face in trying to model any geometric problem is to somehow cut down the number of possibilities something that we can deal with. You cannot deal with infinite numbers.

So, somehow we need to restrict that. So, we need to base on observations. So, let us try to make certain observations about this particular problem navigating a point from the start to the end through these obstacles and when you do that we must also may be just define some kind of a more clean objective function. It is not simply about let us say going from a start to the end point, but let us say we would like to economies something. You know may be we would like to economies space, sorry economies time or the cost of travel somehow that. So, let us say. So, in the shortest path probably may try to we have actually a weights on the edges and we try to seek a path that will minimize the sum of the weights on the edges of the path.

So, similarly here we can think about the solve trying to solve the shortest path problem between start to end and the national notion of the distance here is equidistance. So, whatever is the path, I will just consider that well I mean strictly speaking, you have to prior define measures because it may be it can be any kind of path, but you understand what I mean. Basically the total length of the path is what we would like to minimize with. That you could go to do of the object function. The other objective function could be that I really do not care as long as it is some path, but here if you can let us try to solve the some more interesting problem, we want to actually minimize or find the shortest path find shortest path .So now, it becomes a better defined problem.

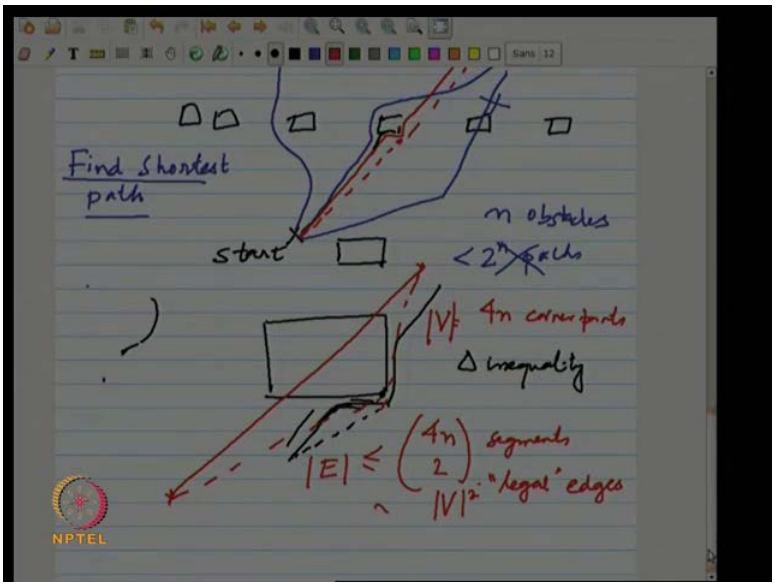
And when you talk about shortest paths (( )). So, the shortest path between two points in a Euclidean space is what. A straight line segment right, well great. So, at least we are getting somewhere, but the straight line segment between this probably passes through or collides in to some obstacles. So, that does not seem like the solution to this problem, but it is a good starting point. So, this red line is not possible, but then we would like to be as close to this is possible. That that is basically what we give as a shortest path in tutorial ((

)) shortest path and if that is so, then one way to think about it is that, suppose this red segment is like a rubber band that we can bend around the obstacles.

So, if it is allowed to bend, then this you know it would not pass through the objects, but my first attempt could be I bend it something like this. So, there are these two bends and that seems to set of be fairly close to the straight line, but do you think this is a shortest path why. Because that shortest path between the starting point and the first bend point will be a straight line (( )).

Great. So, is this an intrusion you use basically because you live in the geometric world? You know exactly it is not like if I live in a graph world, you would not be able to figure out, but you were in the geometric world.

(Refer Slide Time: 21:21)



So, to quickly figure out that something like something like you know. So, this bend, let me blow this up. So, if I blow that up, it is like you have a rectangle and we are talking about a path. So, we do a path like this. Now what you are saying is that, this bend that you took, why did you take this bend because if I take something like this, this the doted should be a short cut of these two bends and why is that?

What is the property that you are using; the triangle inequality exactly. So, you are using triangular inequality. So, you already have a kind of a proof that this dotted line, it

should be shorter than this plus this and therefore, the one that we had before, the solid line cannot be the shortest path.

So, this is another observation that then we will clean up this shortest path that we are trying to find between the start and the end point should have this property that one is that can it consists of curves, can that path have some curves like this.

No same thing, if there is a curve we know that the shortest distance between two points is straight lines I am going to short cut the curve. So, there is no, so, we do not have to deal with any curves. So, we can leave it ourselves to only straight lines and therefore, the shortest path between the starting and the end point is going to be a piece wise linear curve essentially piecewise linear, now, even that piecewise linear, we have further properties that we do not have to look for things like this.

Where are the bends, where can the bends occur the bends can only occur, where, see here there was a bend basically when we collide with an object,, but then that bend is not a bend that should appear in the shortest path the bend can be moved or pushed till it meets a corner point.

Right in other word what I am saying is this red line; let me draw a dotted line again. So, this red line instead of this solid line we should be probably looking for something like this of course, that may crash in to other thing. So, the places that it can bend are only the corners. So, in other words we blew this up you know if I am trying to get from let us say here to here around just suppose we had only one obstacle. So, this straight line when you actually distort and think about it like a rubber band it is going to be basically bend here.

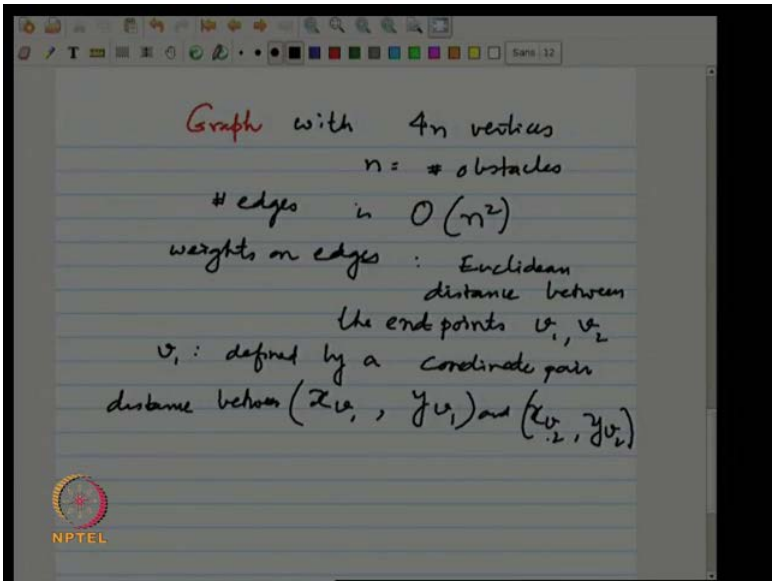
So, the bends only going to be appear at the corner points and we are actually argued in kind of proved it. So, with these two observations in mind now we can (( )) business why. So, now, I can actually model this problem, the shortest path problem as a graph problem where the edges or the vertices are defined by the corner points of the obstacles the four. So, these are rectangular obstacles. So, I can consider the corner points, the rectangles as vertices and the edges must be basically are defined the piecewise linear path. So, the edges must be between two such vertices because there cannot be any other bends, but then there is this thing that not all of them of a legal edges because some. So,

the straight line segment between this and this these two points is not legal between it actually crashes in to an obstacle.

But that again we can verify that you know we will try. So, suppose as I said there are n obstacles, which means that there are 4 n corner points or which I am basically calling the vertices. So, 4 n corner points and then out of the 4 n choose two segments there are some we will define some legal edges that do not crash into obstacles. So, clearly the number of… So, number of vertices. So, if you go back to the graph terminology the number of vertices is 4 n corner points number of edges is bounded by 4 n choose to. So, about let us say about some order n square.

So, if this is or let me use v square etcetera. So, now, you are in the in the graph world and once we are in the graph world some of you find it easier to deal with.

(Refer Slide Time: 26:44)



So, we have a graph with (( )) number of vertices I am saying 4 n vertices where n is the number of obstacles. So, this is a special case of course, we have the obstacles all rectangles essentially the corner points. So, all they have obstacles will be the set of vertices, and the number of edges is less than or let us say I will use big o notation, is big O of n square and you still need another parameter which is basically the weights on the edges.

So, the weights on edges should be what, Euclidean distance. So, the Euclidean distance between the end points let us say v 1, v 2 and every such end point or corner point, v 1 is defined by a coordinate pair. So, there is some x coordinate of v 1 and there is some y coordinate of v 1.

So, when I am saying refer to the Euclidean distance between the end points v 1 and v 2, essentially we are finding the distance between and the corresponding v 2 and y v 2.So the Euclidian distance between these two points should be the weights on the edges. So, there is a formula for the Euclidean distance; the square root of the sum of the difference of the... So, x v 1 v 2 minus v 1 square plus y v 2 minus y v 1 square the square root of that.

And if you do not like square roots you may not even want to do the square roots you compute the shortest paths in this squares because after it is just a relative computation you want to know the shortest path you may not want to know the exact distance. So, we are happy to find a shortest path. So that is it. So now, you are you done your favorite shortest path algorithm on this graph. So, from that space of infinite possible paths you got it down to a very finite description some something like that we are familiar with it in terms of graphs and then you run your shortest path algorithm (( )) algorithm can be done because actually all the distance are non-negative.

So, any questions till now I mean. So, what did we do, we I mean if you want to write a formal proof whatever I have said can be defined in brief. So, what did we argued; we argued that the shortest path between two points without any obstacles the straight line that is the basic fact.

We also argued that the shortest path cannot have curves because curves can always use a shortcut So, therefore, we are only looking at piecewise linear paths the piecewise linear paths where can they bend, they are only going to bend in the obstacles, we are not going to bend without an obstacle because you can gain a shortcut. So, the only point which is going to bend is because of an obstacle, now where is it going to bend; it can bend at an edge or at a corner. So, we proved that it cannot bend in edge because we can actually find a shortest path.

And the moment we restrain two corners then we have only a finite number of paths. So, a shortest path has to be basically comprising of only big line segments. So, the moment

the obstacles are not made straight lines most of the observations would go through, but it will be harder to define what a corner point is. So, then you have to basically deal with. So, 1 is that describing. So, the input to this, what is the input to the problem, I said geometric environment is navigating in geometric environment. So, the input this problem is this geometric environment and the geometric environment has to be described in certain ways.

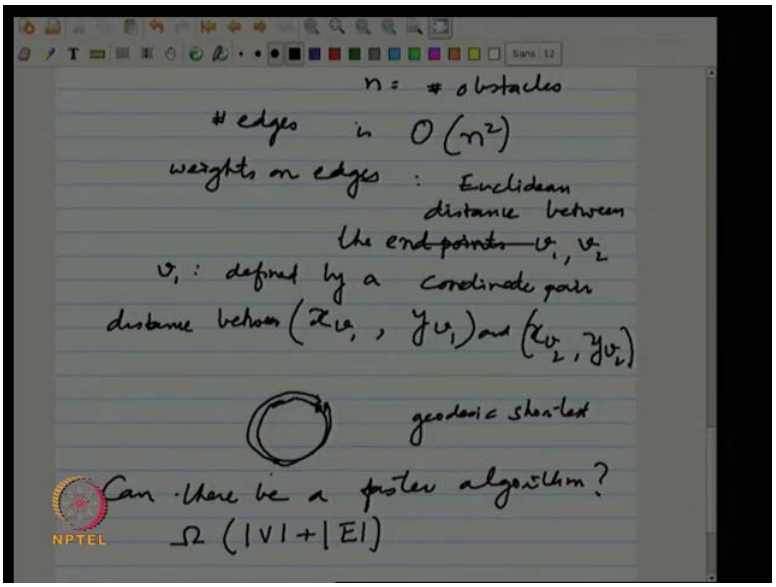
Here if you have only rectangles or rectangular objects the natural description is how you define a rectangle, you can specify all the 4 corners. In fact, you can also specify just the two opposite corners the coordinates of them we are done with it.

If you have objects that have curves, that of surfaces that are the surfaces that are not linear in variety. So, what do you rely on, rely on actually parametric equations.

So, then we have to describe you are seen using parametric equations and then you have to deal with tangents to those you think about you have to deal with tangents to parametric equations. So, also in principle most of the observation that we made we made will go through, but actually implementing when you writing a code etcetera will be much more difficult.

no we still hold right I mean I your point is taking right your point is well taken. So, here what is being said..

(Refer Slide Time: 32:45)



So, here in a special case where you actually have both the points on the surface of the object this is what is called a geodesic path actually? So, this is also very fundamental problem. So, you are on this curve and you have to find the shortest the moment I actually an outside of this, things change.

But this situation that you know I will actually, I have to navigate either like this or like this. And these kinds of problems are called geodesic shortest paths. So, for instance again many of the natural problems like we are on this earth. So, we have if I have to find a shortest path without using an air craft or a space craft. Then I have to basically find the shortest path limited to the surface of the earth. So, this is basically a problem (( )) your point is well taken, in this cases we have paths because you are geo basic means that you have to be on the on the curve, on the surface and therefore, if you cannot leave the surface. So, whatever parametric equation the surface has we will have to you are constrain by that. But these are harder in the sense that these problem. So, I only talked about a solution and the a solution was not that bad in the sense that, if you run to the extras algorithms on this graph which is about n square edges your running time will be about n square. You know n square logged in assumption that it is fairly I mean in the sense that it is it is it is a small polynomial.

But the challenge again is that what we will actually try to do while we develop algorithms no formally is try to develop algorithm. So, the question is can you get something that is faster than n square.

After all your scene is described only by n points. So, let me say let me more preside what I am saying. So, can there be a faster algorithm, this is one of the basic question that we are seeing algorithm course and there is no immediate reason why we cannot do a thing faster than n square.

See when we dealing with a graph you have as input the vertices, the description of the vertices. Description the edges. So, your running time is, cannot be better than something like v plus number of edges.

So, if a number of edges are quadratic, we have to read all the input. So your running time will be quadratic; however, in the case of the geometric problem I were to seen description was using only these, let us say, these four corners of the rectangles which is essentially four end different points..

So, the input is of size of n or 4 n whatever we call it and therefore, the edges that we define are implicit edges, they are not explicitly given to us, the edges were defined, we defined those edges. So, that we could discritize the problems. So, the reason why we got down to n square is because from the continuous space we discritize the problem.

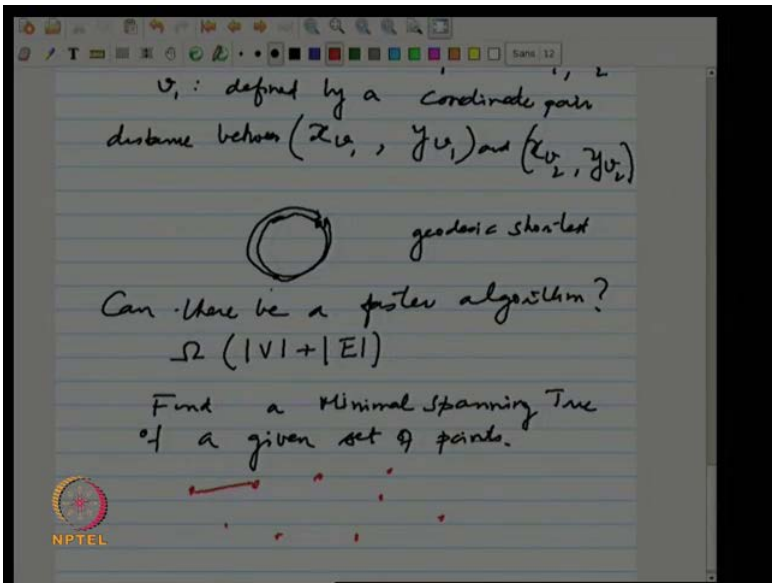
So, there is only some finite number of possibilities on the edges and there could be no more than n square edges, but then who says that we have to find those edges to solve the problem. Maybe there is a more direct way of solving the problem without reducing it to a graph problem, the reason we encountered is n square edges because we try to export it to another frame work that we are familiar with that that is name in the graphs.

And in that process we introduce this edges which could be as many as n square, but perhaps some of these edges may not be required at all and given that we have the geometric information think about some two points like this somehow you know if someone can argue that the starting point is here, the ending point is here, my path is never going to live this rectangle.

Why should we need this rectangle we should be able to know all of that. So, we may be able to actually neither prove nor optimize the search process. Using more geometry which I did not at this point, we use just the minimal amount of geometry required for me to discritize the problem.

So, there could be a first algorithm and this is precisely what we will looking at right without getting with too much details.

(Refer Slide Time: 37:30)



Suppose I were to find a minimal spanning tree of a given set of points, suppose this is a problem given to us. So, we are given this points on the plane, but we have only given this points, we are not actually given the descriptions of the edges is implicit, it is not expressly given to us, we know that any two points can be joined by an edge.

That's all. Which means that could be about end choose two edges now same thing you could run a no graph algorithm on this implicit graph where you have n vertices and n choose two edges and the distance between the weight of an edge is basically distance between the two end points.

And then you cannot do anything better than n square because you have n square edges you have already tied you has by saying that I mean I am going to introduce this edges because I like solving problems in graphs, but then actually that has killed you in the sense that you do not have left any room for improvement.

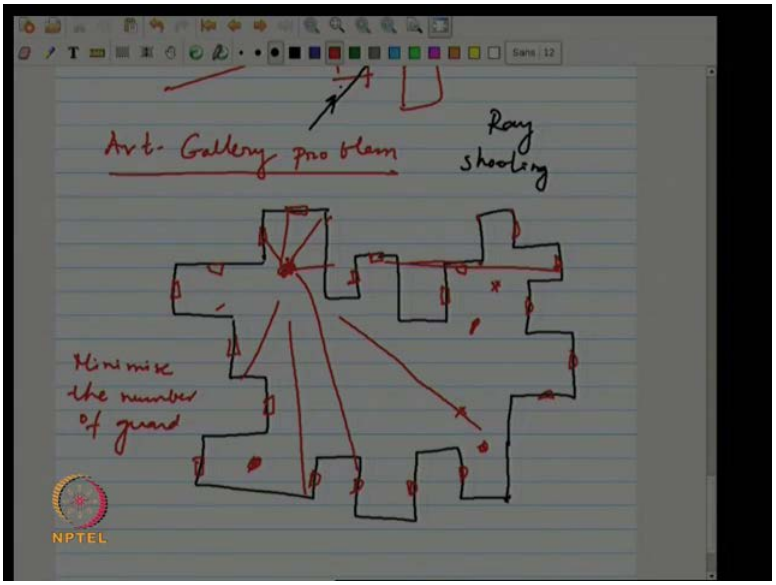
Now, let me tell you that this problem can or has been solved directly without deducing it to a graph right away. So, like we have argued previously that for the shortest path problem you only mean to look at those edges that bend at the corners.

Here again from this n choose two edges, implicit edges we can actually define just about let us say some 3 n edges now, which of these edges would be carried at edges something we will actually look into in details later on when we try to solve this

problem, but there is a way by using geometry we can we could actually reduce from this space of n choose edges to 3 n edges. So, once we reduce it to be about order n edges then, even if you run the graph algorithms the mystery algorithms, it is still close to be linear. So, that is where exactly where you have interplay between the geometry or the using some domain knowledge and also you all with techniques.

So, while introducing this problem actually the shortest path problem is in the general area of what is called visibility problems. So, let me introduce another problem today and we look at the solution perhaps tomorrow in detail.

(Refer Slide Time: 40:13)



So, what I am saying is that there is a important class of problems called visibility problems. Why I am calling visibility problem why is a shortest path (( )) in a visibility problem. So, visibility problems like if I shoot a ray. So, imaginary ray and there is an object, an opaque object the ray cannot cross this like our usual notions of transparent to opaque.

So, this itself is a problem I could give you a scene with many objects and I could specify, here is a ray I am shooting. So, here is the ray I am shooting, now which is the object that this ray will hit first.

So, it is called a ray shooting problem, as such the problem does not appear to be difficult because I can that the ray is of a straight line you know the objects as. So, kind

of description and I can find out you know how this ray and so, which where if this ray is going to intersect what are the points it intersect and then we can find out which is the first point intersects.

So, this is would be a simple straight forward solution, but in the in the for the actual ray shooting problem, what we looking for is that, we should we you know given a scene, we should build some kind of data structure first that when we do when we shoot a ray, you should able to tell me very quickly like let us say login time was something which object there is going to hit. So, that is what makes a problem challenging. Now what is got what a shortest path got to do with it. So, again if you go back to this shortest path problem, it is not very different from the visibility because it's again this like a ray shooting where we are not allowed to pass through these obstacles.

So, the principles used for solving shortest path problem are not very different from the whole class of problems called a visibility problem. So, let me describe a fairly interesting problem. So, let me do first define the problem and then I will tell you the name of the problem. So, we have a large hall and the hall is not a just a rectangle, but has a some corridors and some rooms, but they are they are… So, right now let us assume this is a rectilinear here.

It looks like what one can build. So, you can think about it like a museum or an art gallery I can of that. So, we have look here I am drawing only the projection on the on the on the plane. It is a three dimensional hall huge hall and then as I said this, we thought about as some kind of museum or a gallery and we has exhibits and the exhibits are posted on the walls.

So, there are some exhibits on the walls and we can think about them like paintings on the walls. So, they are these paintings displayed. Let us also assume that these paintings are really expensive, the ones done by (( )). So, they have to be you know they are they are really precious for us. We have to make sure that is even we know we have to guard them properly, that is and there are people because they this… This is a very big business actually write them in (( )) of arts it is a very big business in a people of steal them and they do not have any compassion about that.

And then you go and actually auction them. Even after stealing them you can auction them openly that is what happens so there. So, these paintings have to be really and this is also very common plots for many movies exception.

So, these have to be guarded. So, we have posted some guards in the hall. So, these guards are someone posted here and for a moment let us make this is assumption there no these guards can sort of… So they do not have peripheral visions. I mean they can really sort of look around. So, if a guard is posted here, it can really sort of you know look at all these things simultaneously.

So, it is bit of a in a simplification, but just to state the problem. So, it can keep a watch all these things, but somehow this guard would not be able to keep a watch on this painting because we draw a straight line, the vision is of a straight line. We draw a straight line vision this guard and this painting it involves the walls that it is not visible. So, it goes through some walls. So, it cannot really keep a watch on this particular painting. Your question is that given this scenario, how many guards to be need to place in this whole so that at any point of time, there must be at least one guard that is able to look of your paintings.

So, for every painting, at any point of time, there must be some guard which can see directly. So, one guard here putting one guard here will not suffices because there are many paintings that you are occluded from the guard, but may be if I put a guard here and a guard here and a guard here and a guard here, may be that suffices.

Or you can argue somehow. So, essentially, if you if you draw a disc around these points and the total coverage of this is basically all the walls, then essentially you have you are able to actually keep a watch on these paintings and again the assumption is that if it is been watched, no one will be have to steal it. So, this problem is known as an art gallery problem. So, let me just conclude by saying that this problem, we would like to solve in a fashion where we want to minimize a number of arts.

So, finding a solution is very easy. So, finding a solution for instance, no one can put a guard in every corner and make sure that a no. So, everything is covered. So, it is a covering problem, where you want to minimize the number of guards. So, we will resume from this point from tomorrow.